ADST-II-CDRL-HLAAPPS-2000085
April 3, 2000

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

HIGH LEVEL ARCHITECTURE: APPLICATIONS FOR THE SPECIAL OPERATIONS FORCES

DO #0081

CDRL AB02

FINAL REPORT AND IMPLEMENTATION RESULTS

VOLUME II

For:

United States Army
Simulation, Training, and Instrumentation Command
12350 Research Parkway
Orlando, Florida 32826-3224
Attn: (from DD 1423)

By:

Science Applications International Corporation
12479 Research parkway
Orlando, FL 32826-3248

Lockheed Martin
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	03 APR 2000	FINAL

4. TITLE AND SUBTITLE
Advanced Distributed Simulation Technology II (ADST-II) High Level Architecture: Applications for the Special Operations Forces Final Report & Implementation Results, Volume II

5. FUNDING NUMBERS
N61339-96-D-0002

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lockheed Martin Information Systems
ADST-II
P.O. Box 780217
Orlando Fl 32878-0217

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST-II-CDRL-HLAAPPS-2000085

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32328-3224

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
CDRL AB02

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)* The High Level Architecture Applications to the Special Operations Forces project was a delivery order completed by the Lockheed Martin Advanced Distributed Simulation Technology II (ADST II) integrated product team in an effort to advance the Special Operations Forces' simulation capabilities. The project specifically involved the incorporation of the new DoD High Level Architecture (HLA) into the AC-130U simulator and testbed of the Battle Management Center (BMC) located at Hurlburt Field, FL. The project was sponsored by the U.S. Special Operations Command and administered by the U.S. Army Simulation, Training, and Instrumentation Command (STRICOM).

The Special Operations Forces HLA project consists of a two phase effort. The first phase was completed and involved a requirements analysis, an initial implementation of a Runtime Infrastructure connectivity solution and design, and an execution of a preliminary HLA federation. In addition, the effort also involved the development of a preliminary Special Operations Forces Federation Object Model (FOM). The second phase of this effort involves further work to develop this FOM, but also to continue to implement the HLA migration plan developed under phase one.

14. SUBJECT TERMS	15. NUMBER OF PAGES
STRICOM, ADST-II,	53
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Document Control Information

Revision	Revision History	Date
	Original release	4/3/00
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		

TABLE OF CONTENTS

List of Figures

List of Tables

EXECUTIVE SUMMARY

The High Level Architecture Applications to the Special Operations Forces project was a delivery order completed by the Lockheed Martin Advanced Distributed Simulation Technology II (ADST II) integrated product team in an effort to advance the Special Operations Forces' simulation capabilities. The project specifically involved the incorporation of the new DoD High Level Architecture (HLA) into the AC-130U simulator and testbed of the Battle Management Center (BMC) located at Hurlburt Field, FL. The project was sponsored by the U.S. Special Operations Command and administered by the U.S. Army Simulation, Training, and Instrumentation Command (STRICOM).

The Special Operations Forces HLA project consists of a two phase effort. The first phase was completed and involved a requirements analysis, an initial implementation of a Runtime Infrastructure connectivity solution and design, and an execution of a preliminary HLA federation. In addition, the effort also involved the development of a preliminary Special Operations Forces Federation Object Model (FOM). The second phase of this effort involves further work to develop this FOM, but also to continue to implement the HLA migration plan developed under phase one.

In keeping with this two phase approach, the requirements analysis has been divided into a two volume set. Volume I pertains to phase I requirements analysis and initial implementation efforts. Volume I contains the design analysis of the HLA migration solution and initial implementation phase. It should be clearly understood that Volume I does not represent the final HLA implementation on the BMC training devices. Volume II focuses exclusively on the final delivered HLA implementation. Volume II does not include or make reference to any implementation analysis or phases of implementation prior to final design and implementation.

Fundamental differences between initial HLA implementation detailed in Volume I and final HLA implementation detailed in Volume II are as follows:

- Volume I (initial implementation)

  RTI version 1.0.3

  SOF FOM created and identical to RPR-FOM version 0.3

  The MäK Technologies VR-Link 3.2 product

  STRICOM Gateway 2.3

- Volume II (final implementation)

  RTI version 1.3v6

  SOF FOM created from RPR-FOM version 0.5 with extensions for light state attributes (for SOACMS federates) and IFF class (for Talon federates)

  The MäK Technologies VR-Link 3.4 product

STRICOM Gateway 3.2

Integration of MäK Technologies HLA Data Logger

Integration of MetaVR HLA Stealth Viewer

The requirements analysis involved a study of the BMC training devices to develop a migration plan to convert these devices from the Distributed Interactive Simulation (DIS) protocol to become HLA interoperable. The results of the analysis recommended the incorporation of an HLA middleware solution to the AC-130U simulator. As part of the plan, the other devices of the BMC, including the DIS radios, ModSAF, Stealth, and other DIS devices were connected to a DIS-HLA gateway to permit interoperability with the HLA AC-130U simulator.

The MäK Technologies VR-Link 3.4 product was integrated into the AC-130U as part of the implementation of the RTI Connectivity Solution. In addition, a Simulation Object Model (SOM) was developed for the AC-130U and an initial FOM was generated. Both the SOM and FOM are based on the Real-Time Platform Reference FOM (RPR-FOM). The BMC ModSAF and other DIS devices achieved RTI Connectivity through the use of the STRICOM Gateway. The AC-130U simulator and the STRICOM Gateway both utilize RTI version 1.3v6 and the RPR-FOM version 0.5.

Compliance testing of the final HLA implementation of the AC-130U simulator was successful, with the simulator passing HLA Compliance Testing on June 22, 1999.

Preliminary work was also completed as part of defining the Special Operations Forces FOM. As future HLA products permit the incorporation of unique FOMs, and the need for the Special Operations Forces to advance the simulation capabilities over that defined within the RPR-FOM becomes evident, this effort should continue. In addition, as HLA versions of the COTS DIS products of the BMC become available, the HLA products should be incorporated in place of the DIS products.

The Special Operations Forces have achieved a significant milestone in the HLA compliance of their AC-130U simulator. The full benefits of HLA and distributed simulation will only become realized once other Special Operations Forces training devices have been migrated as well. Once incorporated, the HLA will allow the Special Operations Forces the ability to cost effectively construct highly complex and realistic simulation applications.

1. Introduction

The High Level Architecture (HLA) has been established by the Department of Defense (DoD) in an effort to create an infrastructure that will support simulation reuse and interoperability within the Modeling and Simulation community. The intent of the HLA is to reduce the cost and time associated with constructing individual simulators and groups of interacting simulators, as well as provide an infrastructure that will permit the development of highly complex simulations.

The Special Operations Forces (SOF) have realized the need and benefit of HLA, and have taken a leadership role in pursuing the use of this new technology for distributed simulation. Their support of the HLA has led to HLA certification of the AC-130U Training Device of the Battle Management Center (BMC) located at Hurlburt Field, FL. The migration of the BMC from Distributed Interactive Simulation (DIS) to HLA has paved the way for the Special Operations Forces to fully realize the potential of simulation and distributed simulation technology.

2. Purpose

The purpose of this final report is to document the ADST II effort in supporting the HLA migration effort of the AC-130U training device/test bed. Volume I of this report fully documents the analysis of migrating the legacy devices, the development and integration efforts associated with the HLA implementation, and lessons learned throughout the migration effort. Volume II of this report documents the final HLA implementation configuration of the AC-130U as delivered for training.

3. Contract Overview

In the Statement of Work (SOW), this program was defined to consist of a two part effort. Part 1 was to develop an HLA migration plan for the AC-130U testbed, develop the initial implementation of the migration plan, and develop a general SOF Federation Object Model (FOM). Part 2, conceived as either an add-on or a follow-on effort, was to be accomplished after the conclusion of Part 1. Part 2 was defined to implement the next step of the migration plan developed under Part 1 for the AC-130U. Part 2 was also to develop an HLA migration strategy for the MH-47E and MH-60K simulation systems located at Ft. Campbell, Kentucky, and develop the initial implementation of that strategy.

The SOW defined the overarching objective of this two part effort as an implementation of a SOF Federation using the Federation Development and Execution Process (FEDEP). The SOW defined the following objectives for Part 1:

- Conduct a requirements analysis to determine the best HLA migration path for the AC-130U simulator.

- Develop and execute a detailed implementation plan for AC-130U testbed HLA compliance and testing in accordance with the HLA Rules, Interface Specification, and the Object Model Template (OMT).
- Identify the functional requirements for the development of a Simulation Object Model (SOM) for the AC-130U simulator.
- Design an HLA Federation in accordance with the HLA FEDEP Model that meets the needs of the SOF community and is based upon a SOF mission scenario approved by the SOF community.
- Develop the necessary Simulation Object Model and Federation Object Model in accordance with the HLA OMT that are required to execute an AC-130U exercise.
- Implement a Runtime Infrastructure (RTI) connectivity solution based upon the results of the requirements analysis effort and in accordance with the HLA Rules and Interface Specification.
- Coordinate with other HLA efforts (HLA Integrated Development Environment (HIDE), DO 61; Core DIS Facility (CDF) Upgrade, DO 80; AC-130U DO 02 and DO 78) to prove the value of HLA tools, infrastructure and process in an integrated development activity on ADST II.
- Provide a transition to the efforts in Part 2.

The SOW specified that the following assumptions were to be used in planning this effort:

- Subject Matter Experts (SMEs) will be active participants to expedite identification of the functional requirements, evaluation of designs and review of HLA products.
- All efforts associated with development activities will be conducted at the ADST II base facility in Orlando, Florida, and at the 19th Special Operations Squadron (SOS) at Hurlburt Field, Florida.
- The Government will assist in the allocation of time and priority in scheduling any development efforts required to support the efforts of the program.
- An Integrated Product Team (IPT) approach will be used during the effort. The IPT will consist of contractor, military, academia, and government participants and will continuously leverage from the efforts on other delivery orders including HIDE, DO 61; AC-130U DO 02, DO 52, etc.; CDF Upgrades, DO 13, etc.).
- All software products and COTS hardware will be screened for Year 2000 compliance issues.

The SOW also specified that the following Government Furnished Property/Equipment would be provided on an as required basis at the ADST II Orlando facility by the Government to support the execution of this program:

- ADST II developmental systems from DO 61 and DO 02.

- ADST II HLA Testbed resources from the CDF Upgrade 98 effort.

4. Technical Overview / System Configuration

Two HLA testbed configurations were developed during the course of the SOF HLA program. The OSF HLA Testbed was assembled for CDF upgrade efforts to assist in the HLA development, test and integration efforts prior to on-site integration. The target testbed development for this program, however, was the AC-130U Testbed (BMC) at Hurlburt Field. A discussion on interoperability is also provided to address the requirements of other simulators wishing to participate with the AC-130U simulator in an HLA exercise.

4.1 OSF HLA Testbed Configuration

The majority of the test and integration of the AC-130U software development was performed at the OSF prior to on-site integration. The OSF HLA testbed was assembled under the funding of Delivery Order 80 (CDF 98). The main goal of the CDF 98 HLA effort was to investigate and evaluate HLA products that would be potentially used to migrate CDF sites from DIS to HLA. The testbed infrastructure, along with the system expertise gained as part of the setup and evaluation effort, positions ADST II to support current HLA delivery orders as well as attract future HLA work.

As part of the CDF 98 effort, a survey was performed of currently available HLA products that would be installed in the testbed. These products are described in section 5.6. The results of the survey were used in establishing the OSF HLA testbed, illustrated in Figure 1.

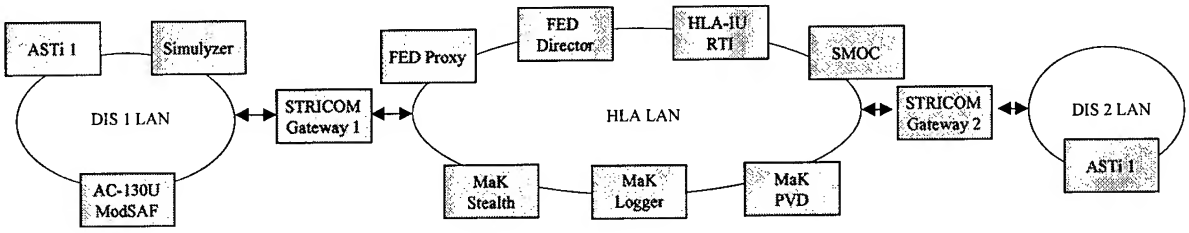

Figure 1 ADST II CDF HLA Testbed

4.2 AC-130U Testbed (BMC) Configuration

4.2.1 Network Diagram

The network diagram of the BMC HLA configuration is illustrated in Figure 2. Note that the DIS network is separated from all HLA traffic. The STRICOM gateway machine (Gateway, formerly referred to as Stealth as it originally hosted the Tasc stealth application) is a dual ported (2 ethernet card) workstation, with one port dedicated to the HLA network and the

other port dedicated to the DIS network. The STRICOM gateway is connected to the HLA-IU through either a standard passive hub or a null modem cable. The hub is used primarily to facilitate the installation of other HLA devices such as testing tools to the HLA network.

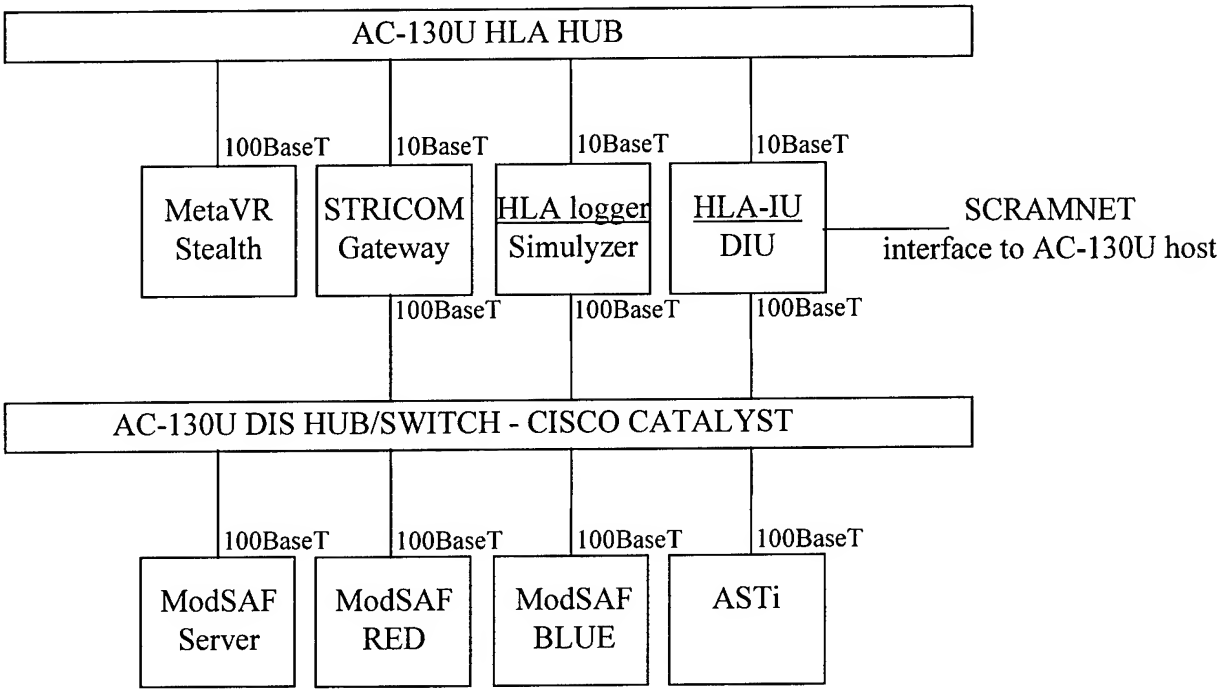

Figure 2 Battle Management Center (BMC) Network Diagram

4.2.2 Statement of Interoperability

The inherent flexibility offered by HLA is intended to achieve a higher degree of reuse and interoperability among participating federates. Much of this flexibility results from the ability for federate developers to define the information model contract (known as the FOM) for a particular federation execution. However, to be able to interact with other federates and achieve interoperability, it is essential that each federate comply with the data contract defined in the FOM. Failure to comply may result in data communication problems between federates, and a less interoperable federation execution. It is also important to note that achieving proper data interactions between federates is necessary but does not guarantee interoperability, as other issues such as database correlation must also be addressed.

The AC-130U utilizes the Special Operations Forces (SOF) Federation Object Model (FOM) which is based off of the Real-Time Platform Reference FOM (RPR-FOM). For another federate to interact with any federate in the SOF federationthat federate (simulator) must support the data contract defined in the SOF FOM. For specific, detailed capabilities of the AC-130U, federate developers must look at the AC-130U Simulation Object Model (SOM) to determine detailed interoperability. The SOM is a subset of the FOM, and the FOM

encompasses the entire set of objects and interactions that can be exchanged by the current list of participating SOF federates.
While the above discussion implies a complex approach to determine simulator interoperability within a federation, this process is simplified considerably by the SOF FOM's similarity to the RPR-FOM. Other federates using the same version of the RPR-FOM utilize many of the same class objects and interactions, attributes and parameters, datatypes, etc. within their SOM. Section 6.2.3 and appendix F of this report describe the AC-130U FOM as compliant to a variant of RPR-FOM version 0.5.

Another issue worth noting relating to compatibility involves the use of the RTI. It is imperative that all federates desiring to interact with the AC-130U in a federation execution utilize the same version of the RTI. In its current configuration, the AC-130U and the STRICOM gateway both utilize RTI version 1.3v6.

5. Technical Approach

5.1 RTI Versions

RTI 1.3 version 6 has been used for the final delivered HLA implementation. The HLA-IU as well as all the associated HLA tools (MetaVR Stealth, MaK Data Logger and STRICOM Gateway) all utilize this RTI version.

5.2 RPR-FOM Versions

The RPR-FOM was selected as the target reference FOM for developing the AC-130U SOM and SOF FOM. The RPR-FOM is a reference FOM developed to meet the needs of the Real-Time Platform and former DIS community. The RPR-FOM was primarily chosen because it was the only FOM supported by the Commercial Off The Shelf (COTS) HLA products (including the gateways and middleware solutions) considered as part of the migration plan. In addition, as the AC-130U was a former DIS (version 2.0.4) simulator, the use of the RPR-FOM was recognized as the most efficient approach to achieve an HLA solution.

RPR-FOM version 0.5 with extensions (see appendices) is being used for the AC-130U HLA implementation.

5.3 RTI Connectivity

5.3.1 Results of Analysis

Following the analysis of the different approaches to HLA migration, the IPT developed a migration strategy for the DIS devices of the BMC. This strategy included the integration of a middleware product to the AC-130U, replacing the DIU with an HLA Interface Unit. This solution was the most cost effective HLA-based solution, and was achievable within the time-frame of the contract.

The middleware solution involves the integration of a commercially available software product to the simulation device that handles all interfaces to the RTI and network. This product typically comes with an API that the software developer uses to link the simulator application to the middleware software. The middleware solution offers low latency, scalability, and potentially full HLA capability. In addition, the middleware solution can offer forward compatibility (to newer HLA versions), and backward compatibility (for DIS simulators).

For all other DIS devices of the BMC, it was decided to utilize a gateway to convert between DIS and HLA. These DIS devices included ModSAF, ASTi DIS Radios, Stealth, PVD, Exercise Manager, and Data Logger (TASC Simulyzer).

5.3.1.1 MäK Technologies VR-Link

The MäK Technologies VR-Link version 3.4 product was chosen as the middleware solution for the HLA-IU. VR-Link is referred to as a Protocol Interface Unit (PIU), as it includes an API that supports both the HLA and DIS protocols. The main advantage of using VR-Link 3.4 was that the current DIU utilized a previous version of VR-Link (2.4.5), thus facilitating an easier integration approach. In addition, as a reputable commercial product, VR-Link offers a maintenance agreement which provides technical support and software upgrades to support newer versions of the RTI and RPR-FOM as they are released. The VR-Link software with maintenance agreement also proved to be cost effective compared to the labor investment required for the SMOC software. For these reasons, VR-Link was selected as the middleware solution for the HLA-IU.

5.4 Gateway Solution

5.4.1 STRICOM Gateway

The STRICOM gateway was chosen as the gateway solution for the BMC. The STRICOM gateway offered several advantages, including full support of the most current version of the RPR-FOM and good technical support. In addition, others in the HLA community had also attained success using the STRICOM gateway as their solution. Also, future releases of the gateway are alleged to be FOM flexible, allowing the use of different FOMs. For these reasons, the STRICOM gateway was selected as the HLA gateway solution for the DIS native COTS devices of the BMC.

5.5 CGF Survey

A survey was also conducted as part of the analysis to determine the suitability of the BMC ModSAF and potential HLA alternatives. The BMC has a modified version of ModSAF 3.0 that runs DIS 2.0.4. ModSAF, a semi-automated Computer Generated Force (CGF), is used as a target and threat generator. This CGF system has the ability to model the physical and behavioral representation necessary for soldiers-in-the-loop not to be able to distinguish simulations of human behavior, missile flyouts, and communications from other soldier-in-the-loop simulators and live instrumentation. Players make doctrinal and tactical decisions, deploy forces, develop scenarios, and make and execute plans. CGF is a fundamental

building block of Advanced Distributed Systems (ADS) technology because it is the only economically practical method of populating the ADS environment.

The HLA Gateway provides protocol translation between HLA services and ModSAF. For reasons discussed in section 5.4, the gateway approach was the best solution for integrating the DIS ModSAF into an HLA environment.

6. Technical Implementation / Lessons Learned

6.1 RTI Connectivity

RTI Connectivity was accomplished through the use of the MäK Technologies VR-Link Middleware (version 3.4) software. VR-Link 3.4 provides RTI version 1.3v6 connectivity and compliance to HLA Interface Specification version 1.1.

6.1.1 Host Emulator

6.1.1.1 Introduction

The purpose of the Host Emulator was to provide a means to simplify and speed up testing of the HLA-IU software. The emulator will do this by placing data into the appropriate memory locations, as if it were the actual AC-130U simulator host accessing SCRAMNet. The term "SCRAMNet memory" is used throughout this text to refer to a structured shared memory area that two or more processes can access to transfer information (such as various data elements).

Additional functionality is provided by a primitive flight simulation routine. This routine provides constantly updated ownship location/movement information to the appropriate memory locations, in such a manner as to simulate a simple, circular, flight path.

6.1.1.2 Software Operation

There are two primary aspects of the AC-130U Host Emulator. The first is the Graphical User Interface (GUI), which is activated when the user enters the program name "emulator" at the UNIX prompt (and while in the appropriate directory). The second is the Initialization file, which will generally be constructed in the same directory.

The GUI consists of 5 distinct regions: Title Bar, Screen Selection Area, Data Entry Area, Command Button Line, and Message Area. In addition, a copy of the operator's manual is provided as a help text popup area. Each of these areas, and the capabilities of each, are described in the operator's manual.

A capability for reading initial values from a text data file has been included. To utilize this capability it is first necessary to create the text file, consisting of a list of variable names (as seen on the data entry screen) with an associated value. More details are available in the online operator's manual.

6.1.1.3 Software Overview

6.1.1.3.1 Purpose

In order to decrease development time of the emulator, and to place the Emulator development effort into the realm of the DoD's software reuse initiative, the data object identification portions of the existing AC-130U software were used. In the event that this software may be reused for future efforts, a brief overview of the functionality is provided here.

6.1.1.3.2 Functional Areas

6.1.1.3.2.1 Graphical User Interface

To facilitate ease-of-use of this software, the user interface was written in an X-Windows style, using Motif & X-Toolkit meta-calls. All of the XWindows setup is performed in the "main" and "add_???" routines. The series of "CB_????" (CallBack) routines are used to process any mouse or keypad inputs to the GUI area.

In order to rapidly switch between the Data Entry Areas (when one is selected in the Screen Selection Area), it was found that the easiest method was to pre-create each screen at startup. Then we would just use the XtManageChild & XtUnmanageChild calls to make only one screen active at a time.

6.1.1.3.2.2 Emulator to HLA-IU Interface

In the Hurlburt configuration, the SCRAMNet Shared Memory system is used to pass data between the AC-130U Host computer and the DIS (now HLA) Interface Unit computer. For our purposes, we are using the UNIX interprocess shared memory functions ("shmget" and "shmat"). In order for this to work, both our Host Emulator and the HLA-IU must be running on the same computer. The first process running will allocate a block of shared memory, and will write the address of this block to a text file, so that the second process can attach to it.

To avoid overflowing the UNIX memory with unreleased blocks, a destructor routine was added that removes the shared memory block (using "shmctl") and the "scraddr.txt" file, whenever one of the processes exits. Because the memory is not actually released until all attached processes become unattached, it is necessary to be sure both processes are down, before restarting either. Otherwise, restarting the down process will cause a new section of UNIX shared memory to be allocated, when it doesn't find the "scraddr.txt" file.

6.1.1.3.2.3 Ownship Movement

The ownship movement algorithm is implemented by taking a circle and dividing it by the number of desired iterations, and determining the position of a point on that circle for each iteration. The only non-intuitive aspect of this functionality is that the heading of the vehicle (OWNYAW) appears to be based as if the vehicle was traveling backwards, in reverse [2π - (angle from circle origin to vehicle)].

6.1.1.3.2.4 Initialization Data File

The initialization data file is implemented by reading in a text file, and interpreting and writing the data items to the shared memory. The "process_data_to_mem" function is written in a generic manner so that it can be used from several areas.

6.2 Object Model Development

The HLA requires that all federations and federates are described by an object model that identifies all data that is communicated between federation members during an exercise execution. HLA federates are described by a Simulation Object Model, while federations are described by a Federation Object Model. The HLA Object Model Template provides the common format and syntax used to document both the SOM and FOM. The OMT consists of an object model identification table, object class structure table, interaction class structure table, attribute table, parameter table, and FOM/SOM lexicon.

6.2.1 SOM Development

The HLA requires that all federates have a SOM, documented in accordance with the HLA OMT (Rule 6). The SOM is a specification of the intrinsic capabilities that a federate can offer to an HLA federation, and is used to determine the suitability of a federate to participate in an exercise.

There exist several approaches to SOM development, all of which may produce different yet accurate results. One approach is to follow the bottom-up approach outlined by Robert Lutz at the 1997 Spring Interoperability Workshop, and document via the OMT all objects, attributes, interactions, and parameters that are defined by the internal model of the simulator. This data would then be organized in such a fashion to best represent a simulator's desired publication and subscription capabilities within a potential federation.

However, if the federation has already been defined and a FOM or reference FOM exists, the above approach may be simplified. This was the case with the AC-130U, in which it was decided to utilize the Real-Time Platform Reference FOM. In this situation, the simulator's SOM will follow closely the object classes, interactions, attributes, and parameters as well as the object and interaction hierarchies of the FOM. The SOM will appear as a subset of the FOM, identifying its capabilities within the federation.

The AC-130U SOM was created by identifying the simulator's publish and subscribe capabilities for all object classes, interactions, attributes, and parameters within the SOF FOM. This was done essentially by mapping the data within the SCRAMNet shared memory block of the AC-130U DIS Interface Unit (DIU) to the respective elements of the SOF FOM. The SOM was then constructed by utilizing the Object Model Development Tool (OMDT) version 1.3v3 provided by the Defense Modeling and Simulation Office (DMSO). The RPR-FOM version 0.5 model was used as a baseline, and any elements that were not supported by the AC-130U were eliminated. In addition, the EventID complex datatype was

changed to follow the RPR-FOM version 0.5convention. The resultant SOM retained the same class hierarchy as the RPR-FOM.

The class subscription ability of the AC-130U utilized the default subscription ability as implemented by VR-Link. VR-Link utilizes the SOF FOM, and as a middleware product, automatically subscribes to all object classes that represent entities in the SOF FOM, including Base Entity, Physical Entity, and all subclasses. The implication of this is that during federate initialization, the AC-130U declares (using Declaration Management Services) interest in receiving updates for all objects within the federation. Thus, the AC-130U federate is effectively not using Declaration Management to filter any federation data. However, since the intended federation consists entirely of entities that the AC-130U can support and is interested in, this is not a problem. In the future, however, if the AC-130U is to participate in large federations and requires a need to filter federation data, VR-Link allows the developer the ability to restrict subscription capability to desired classes.

6.2.2 SOF FOM Development and Implementation

The SOF FOM development activity for this project focused on development of the requirements for a FOM which is geared to meet the unique needs of the SOF community. Since the SOF FOM was not required for the development of the initial federation, the SOF FOM development activity initiated a process for identifying components of the SOF FOM and provided a plan for development of a general FOM for use by the SOF community.

The SOF federation utilized a variant of version 0.5 of the RPR-FOM as its federation object model, illustrated in Appendix E. SOF FOM extensions to the RPR_FOM 0.5 baseline consist of the addition of light state attributes (for SOACMS federates) and the IFF class (for Talon federates). The SOF FOM is primarily a subset of the RPR-FOM version 0.5 (with extensions previously noted) and was created by using the RPR-FOM as a baseline and removing unsupported object and interaction classes, attributes, parameters and data types. The AC-130U SOM is identical to the SOF FOM with respect to all identified object and interaction classes, attributes and parameters.

Another issue relating to the FOM implementation is the uniqueness of the .fed file used for the SOF federation. The .fed file is used by the RTI during initialization and declaration management functions, and is traditionally generated from the FOM using the OMDT. However, because VR-Link and the STRICOM Gateway both add additional classes to the .fed file for internal federate initializations, the FOM cannot be used to generate the .fed file. The additional classes could have been made as part of the FOM, but that would violate the OMT specification that there should not exist any concrete classes within the FOM that are neither published nor subscribed by anyone. Appendix H defines the differences between the RPR-FOM version 0.5 .fed file and the SOF .fed file.

6.2.2.1 SOF FOM Development Meeting

A critical component of SOF FOM development was to conduct discussions with SOF community experts. A meeting with Major Mike Vaughn was conducted on August 13, 1998. The purpose of the meeting was to explore the need for a special federation object

model for the SOF community. The proposed FOM would eventually serve as a basis for phase II HLA interface development for the AC-130U and its interconnectivity with other SOF simulations. The SOF FOM might eventually serve as a starting point for a SOF community persistent FOM.

As a point of reference, the requirements for the SOF FOM would be based on providing support for the AC-130U testbed assets and the simulation devices at Ft. Campbell. Other devices at Hurlburt Field might also be included. The RPR-FOM, which is currently in use with the initial AC-130U HLA interface, will serve as the starting point for the FOM development activity.

The following questions / issues were posed to identify key requirements for the SOF FOM:

What specialized tactics, such as de-mining operations, Psychological warfare activities, etc. are required for SOF that are not currently supported in the RPR-FOM (and similar federations)?

At a minimum, we need to ensure that the FOM provides adequate representation for various SOF equipment configurations.

Weapons of Mass Destruction (WMD) such as Biological, Chemical and Nuclear represent a new set of weapon effects and representations that may require additional attributes and interactions beyond what is currently provided for conventional weapons.

Does the object class hierarchy of the RPR-FOM serve the SOF community well?

What types of things are needed to support Mission Rehearsal? We need to address data collection, AAR type functions for these areas. This also brings up the need to address tools as well as FOM issues.

Need to review electronic / sensor representations to ensure they provide sufficient support for SOF operations.

Answers for these questions were not identified during the meeting due to the possible classified nature of the responses. A decision was made to conduct a follow-up meeting at Hurlburt Field, possibly during the interface integration phase, to further discuss these details. One product resulting from the meeting was a proposed matrix of SOMs for the various platforms of interest. It was determined during the discussion that by combining the SOMs or capabilities of the systems, we could identify a common FOM that would address the various systems. The matrix, illustrated in Table 2, includes the platform types on one axis and missions on the other.

Table 1. Matrix of SOMs / Requirements

Platforms→ Missions	**AC-130U**	MC-130E WST	MC-130H	MH-60	MH-47	CV-22
Airfield Seizure						
Infill/Exfill						
Helo Assault						
Close Air Support						
Interdiction						
Recon-naisance						
Command & Control						
Escort						

The RPR-FOM was examined to remove elements that were not of interest to the SOF community. This includes DIS legacy items such as Guises. Further discussion of capabilities required, to be identified when Table 2 is completed, will be required to determine other elements of the RPR-FOM that should be removed or augmented.

Due to schedule and budget constraints, follow-up meetings were not conducted and a draft FOM was not developed. Further work in this area is envisioned under anticipated follow-on efforts.

6.3 HLA Interface Unit (HLA-IU)

The HLA Interface Unit is a software application that processes the SOF FOM data. The HLA-IU resides on a Silicon Graphics Inc. Indigo 2 computer separate from the Host, and communicates with the Host through shared memory, consisting of a SCRAMNET connection.

Functionally, the HLA-IU provides two services for the Host, publication and subscription of FOM data. The HLA-IU subscribes to SOF FOM data and places it into shared memory for the Host to read and process. The HLA-IU then reads data from the Host and publishes it as SOF FOM data to the network. All SOF FOM data subscription and publication functions supported are in accordance with RTI1.3v6 HLA specifications.

Custom software makes bi-directional data transfers between the local shared memory and the RTI through VR-Link, which is a COTS HLA API. VR-Link packages outgoing data, dead-reckons entity data, provides smoothing and filtering, and makes most of the RTI service calls to the local RTI component (LRC).

The following diagram shows the concept of placing a HLA Interface Unit (HLA-IU is depicted as the shaded region) between the Host and the HLA network. In accordance with HLA rules, all network communication occurs between the HLA-IU and the RTI.

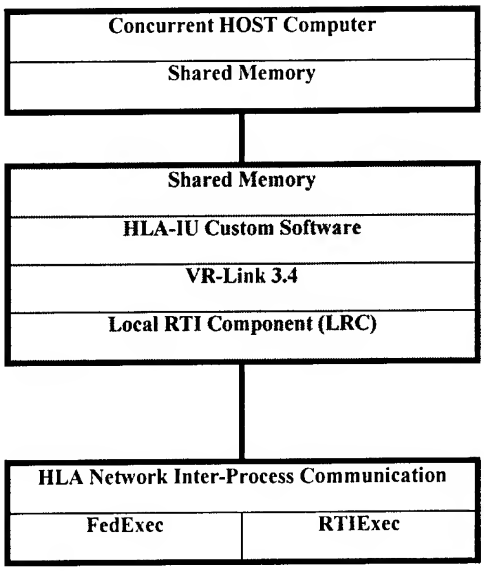

Figure 3 HLA-IU Communications Diagram

6.3.1 RTI 1.3v6

The HLA component of the HLA-IU is the Run-Time Infrastructure (RTI) version 1.3 release 6. The RTI serves as the data distribution mechanism in HLA. It provides communication between the HLA-IU and the Local RTI Components (LRC's) of other applications or federates.

The RTI was downloaded from the Defense Modeling and Simulation Office (DMSO) at http://hla.dmso.mil/. This software package includes installation software, release notes, and several source-code examples illustrating much of the API functionality. This software must be installed on and configured for every federate participating in a federation.

The RTI software was installed in the directory path */usr/people/cm/rti1.3v6.*

6.3.2 VR-Link 3.4

The VR-Link Networking Toolkit version 3.4 from MäK Technologies is an object-oriented library of C++ functions and definitions that are used to communicate between the HLA-IU

custom software and the RTI. The VR-Link 3.4 software was installed in the directory path */usr/people/hlaiu/vrlink3.4.*

The following are VR-Link's features to speed the creation and maintenance of HLA applications:

6.3.2.1 Exercise Connection Class

The DtExerciseConn class is the exercise connection and serves as the application's interface to the RTI. It is the object through which a VR-Link application connects to the network. VR-Link will create an instance of this class, called *exConn.* Through this object VR-Link can send and receive simulation data. This object provides an efficient way to direct incoming data to other parts of the HLA-IU. Its member functions allow an application to send interactions to the exercise, read input from the network, generate event IDs, and register call back functions.

6.3.2.2 Object Management Classes

To better manage state information, VR-Link informs other exercise participants about states of its locally generated objects, such as entities and emitter systems. In turn, VR-Link receives, processes, and manages state information about remote objects. Object Management classes maintain state information of local and remote objects, and handle the sending and receiving of state updates. These classes provide an entity list to keep track of participants in the federation, and facilitate dead reckoning, trajectory smoothing and filtering.

Entity state and network entity data is maintained by the following VR-Link C++ container classes:

DtEntityPublisher, DtEmitterSystemPublisher, DtEmitterBeamPublisher

> These publisher classes send state update messages of its locally simulated objects to other exercise participants

DtEntityStateRepository, DtEmitterSystemRepository, DtEmitterBeamRepository

> These storage classes store the locally or remotely simulated object's state information. They contain functions for inspecting and changing the components of an object's state.

DtReflectedEntityList, DtReflectedEmitterSystemList, DtReflectedEmitterBeamList

> VR-Link maintains a list of remote objects in linked lists. Each object known to the list has a DtReflectedEntity, DtReflectedEmitterSystem, or DtReflectedEmitterBeam object to represent it.

Reference the VR-Link User's Guide for a detailed description of each container class.

6.3.2.3 Interaction Classes

Interactions are one-time events such as weapon fire or munition detonations. VR-Link manages interactions through classes derived from DtInteraction, such as DtFireInteraction and DtMunitionDetonationInteraction.

When an interaction occurs on the network, VR-Link receives notification through the exercise connection. The function DtExerciseConn::drainInput() reads and processes the input. The HLA-IU custom software reacts to the interaction through the use of callback functions.

Interactions created by the Host are passed to the HLA-IU custom software, which packages it using VR-Link container classes, which in turn sends the interaction through the exercise connection.

6.3.3 HLA-IU Custom Software

The HLA-IU custom software is an adaptation of the DIS Interface Unit (DIU) already in place from a previous delivery order. The goal of our development plan was to leave the shared memory data structure intact and convert existing DIS class functions to HLA class function calls. In addition, the developers reorganized the source code for improved reading and use of the C++ programming language.

The HLA-IU custom software was installed in the directory path */usr/people/hlaiu/src*.

6.3.3.1 The HLA-IU Software Process

The following flowchart is a high level diagram of the lifecycle of the HLA-IU custom software. Following the flowchart is a brief description of each step in the process.

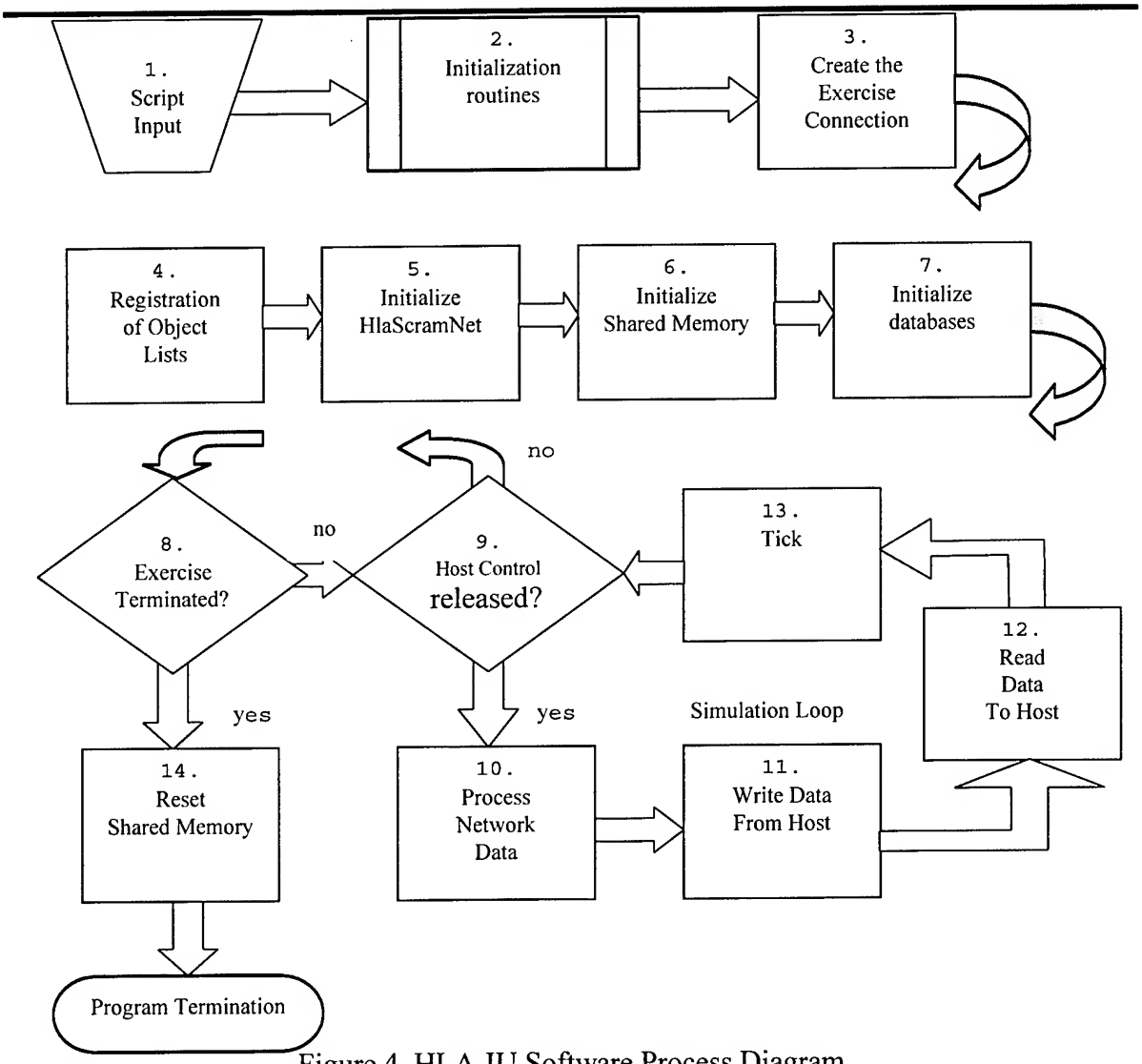

Figure 4 HLA-IU Software Process Diagram

1. Script Input

The HLA-IU custom software is launched from the Toolchest menu. The executable name for the application is *hla*, and is located in the /usr/people/hlaiu/src directory.

2. Initialization Routines

Several initialization routines are called during the startup process. DtTimeInit() initializes the simulation time so that all entities are dead-reckoned based on the same value of current time. InitDebug() initializes a command line input parser menu, which allows the user to view specific network data. InitMtl() reads and processes the runtime data defined in the *mtl* file.

3. Create the Exercise Connection

DtExerciseConn is the object through with a VR-Link application connects to the network. Data is sent and received through the exercise connection. Three arguments are passed to the DtExerciseConn constructor. *ExecName* is defined in the *mtl* file, and refers to the federation name (sofhla). *FedName* is also defined in the *mtl* file, and refers to the federate name (AC130U). The third parameter specifies the version of the RPR-FOM it plans to use(0.5).

4. Registration of Object Lists

The class MyEntityStateRep registers objects it will publish. The object instance *emt_sys_list* of class DtReflectedEmitterSystemList is registered for subscription of Emitter Systems. The object instance *remotes_list* of class type DtReflectedEntityList is registered for subscription of the following object classes:

BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.MilitaryAirLandPlatform

BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.MilitaryLandPlatform

BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.MilitarySeaSurfacePlatform

BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.MilitaryMultiDomainPlatform

BaseEntity.PhysicalEntity.MilitaryEntity.MunitionEntity

BaseEntity.PhysicalEntity.MilitaryEntity.Soldier

5. Initialize HlaScramNet Object

The custom defined C++ class HlaScramNet maintains the process of passing the information between VR-Link and the Host.

The instance *ac130* of class HlaScramNet represents all the properties of reading and writing to shared memory. Private and public member functions process the data through the simulation loop. Several important steps are done during initialization of the *ac130* object, including:

- Pointers to shared memory are allocated
- Range filtering is determined
- Determine which device is represented (readOwnship())
- Site/Host/ID is read and assigned
- DIS enumeration for locally generated AC-130U is assigned
- Entity Publisher is initialized
- Interaction callbacks are registered

6. Initialize Shared Memory

The init_mem() member function defined in the file hla_init_mem.cc clears the following memory locations of data:

- Read/Write flags
- Network Entity Data
- Network Fire Interaction Data
- Network Detonation Interaction Data
- Network Emission Data
- Start/Stop flags
- Ownship Positional Data
- Ownship Fire Data
- Ownship Detonation Data
- Ownship Collision Data

This initialization routine remains identical to the DIU scramnet initialization block. During development an effort was made to clear (zero out) scramnet data by zeroing out from the start address to the end address of the shared memory block. This did not work. The host computer initializes values when it is first loaded, and never writes those values again. Zeroing out the data on an HLA-IU restart would permanently lose the original values of certain host variables (such as ownship emission data). As a result, the development team chose to leave the initialization routine identical to the DIU's.

7. Initialize Databases

Two databases are loaded at startup. The mapping of DIS entity enumerations to image generator models is contained in the file *hlasosvismap.txt*. This file is read and the data is stored into 4 storage arrays (enumeration, model type, model name, and model class). The second database loaded may or may not contain any data. The file *hlafilter.txt* can contain DIS enumerations of entities to be filtered out and therefore not processed by the host computer. The enumerations consist of 8 integers separated by semicolons. The wildcard "-1" will allow any number to match that slot. This file is also read and the data is stored into an array.

8. Exercise State Check

This step in the application is the start of the simulation loop. All data processing occurs after this point. At the start of each frame of the simulation loop, the exercise is polled for termination. If termination has not been cued (by user input of "q" on the parser), processing of data is done for one frame of simulation.

9. Host Control Processing

During each simulation frame, the host must relinquish control to the HLA-IU for processing. If the host does not relinquish control, a counter is incremented indicating the HLA-IU is

alive, and a tick is sent to the RTI through the function drainInput(). If the host relinquishes control, data is processed.

10. Process Network Data

The function drainInput() is the VR-Link function that reads and processes all network data. The simulation time spent inside this function is the time spent by the VR-Link API. The HLA-IU custom software processing spends the remaining simulation time.

11. Write Data To Host

Data is collected from the network during drainInput() and stored in various VR-Link container classes. The data is then retrieved by the following HlaScramNet member functions and sent to the host:

- writeBaseEntAttrToHost – this function is the most time intensive function in the custom software. All network entities are sorted by slant range to determine the closest 88 entities to the host. The sorted entities are then processed so that they are sent to the host shared memory in the same memory slot as for the previous pass. The algorithm is difficult to follow. A simpler implementation using a linked list rather than a bubble sort of an array proved to be more time consuming and was dropped. This member function is defined in the file hla_wr_base_ent.cc.
- writeWeapFireParamToHost – WeaponFire data collected through an interaction callback is sent to the host shared memory. The maximum fire interactions processed per simulation frame are ten. This member function is defined in the file hla_wr_fire.cc.
- writeMunDetParamToHost – MunitionDetonation class data collected through an interaction callback is sent to the host shared memory. . The maximum detonation interactions processed per simulation frame are ten. This member function is defined in the file hla_wr_deton.cc.
- writeEmissionsToHost – Emitter System and Beam class data that maps to an existing entity on the sorted list structure (which is built in writeBaseEntAttribToHost()) is packaged, sorted, and the closest ten are sent to the host shared memory. This member function is defined in the file hla_wr_emis.cc.
- writeStartResumeParamToHost – Two modes of operation exist for the host computer. Integrated mode allows the network start interactions to resume the host operations. In independent mode, the network start interactions are ignored. This member function is defined in the file hla_wr_start.cc.
- writeStopFreezeParamToHost – Two modes of operation exist for the host computer. Integrated mode allows the network stop interactions to halt the host operations. In independent mode, the network stop interactions are ignored. This member function is defined in the file hla_wr_stop.cc.

12. Read Data From Host

Data for various object classes are read from shared memory, and are packaged by the following member functions:

- readBaseEntAttrFromHost – BaseEntity data is read from shared memory and placed into an instance of the DtEntityPublisher class for the ownship. This member function is defined in the file hla_rd_base_ent.cc.
- readCollParamFromHost – Collision data is read from shared memory and placed into an instance of the DtCollisionInteraction class. This member function is defined in the file hla_rd_collision.cc.
- readEmissionsFromHost – Emitter System data is read from shared memory and placed into an instance of the DtEmitterSystemPublisher or DtEmitterBeamPublisher class. The AC130U can publish three emitter systems and each system can generate one beam. Note that some system and beam data values are hard coded because the host does not generate an important piece of data, or in the case of beams, it erroneously generates the same data for all beams. This member function is defined in the file hla_rd_emis.cc.
- readWeapFireParamFromHost – Weapon fire data is read from shared memory and placed into an instance of the DtFireInteraction class. The host can publish up to two fire events per simulation frame. This member function is defined in the file hla_rd_fire.cc.
- readMunDetParamFromHost – Munition detonation data is read from shared memory and placed into an instance of the DtDetonationInteraction class. The host can publish up to four detonation events per simulation frame. This member function is defined in the file hla_rd_deton.cc.
- readActReqParamFromHost – Action request data is read from shared memory and placed into an instance of the DtActionRequestInteraction class. Either of two conditions will successfully send an action request to load a scenario. First, the host must set the flag N_DIU_MODSAF_LOAD_REQ and send a non-empty scenario name. Second, the flag WAITING_TO_SEND_LOAD_REQ is set. This member function is defined in the file hla_rd_action.cc.
- readStartResumeFromHost – Start interaction data is read from shared memory and placed into an instance of DtStartResumeInteraction class. The HLA-IU will only send a start interaction if the host is in independent mode, and HostIsInStopMode flag is set, and the host N_DIU_HOST_FREEZE_REQ flag is not set. This member function is defined in the file hla_rd_start.cc.
- readStopFreezeFromHost – Stop interaction data is read from shared memory and placed into an instance of DtStopFreezeInteraction class. The HLA-IU will only send a stop interaction if the host is in independent mode, and the HostIsInStartMode flag is set, and the host N_DIU_HOST_FREEZE_REQ flag is set. This member function is defined in the file hla_rd_stop.cc.

13. Tick

The member function simTick() is where all object state repositories will send the data through VR-Link to the network.

14. Reset the Shared Memory

During development it was noticed that upon restarting the DIU, entities that were no longer on the network were still viewed on the image generator. It was discovered that the host does not clear network data variables, therefore values remained until overwritten by the DIU. The development team improved operations by calling the same init_mem() function upon exiting the application. (Note: the change was only done on the HLA-IU)

6.3.3.2 HLA-IU Run-Time Configuration Data

The HLA-IU reads and processes configuration files at runtime. These files can be modified without having to recompile the HLA-IU source code.

6.3.3.2.1 MTL files

An *mtl* file is a MaK Technologies Lisp configuration file. This type of file contains default parameters for the HLA-IU. The *mtl* is stored in the mounted */simulation* directory on the Indigo 2 computer. Rather than introducing a second identical file to be read for the HLA-IU as for the DIU, it was determined to read the same *mtl* used by the DIU. The file */simulation/diu.mtl* is read in at runtime and contains required parameters for both the DIU and the HLA-IU.

The following are the selections that are suitable for changing, thus taking advantage of dynamic exercise planning:

ExerciseId	This option changes the exercise id. The default is 10.
own_marking	This option changes the markings of the training device. The default is "Spooky"
AreaOfInterest	This option will take as an integer the slant range in meters that network entities must be from the ownship in order to be processed. Network entities exceeding this slant range will be excluded from processing. The default is 160000 meters, roughly about 100 miles.

The following can be edited, but the HLA integration team discourages this unless the network operator is aware of the implications of making these changes:

ExecName	This option changes the name of the federation. It takes a string of less than 28 characters. The default is "sofhla"
FedName	This option changes the name of the device as listed on the fedEx window. It takes a string of less than 28 characters. The default is "AC130U"
SiteId	This option changes the site id of the training device. The default is 26.
ApplicationNum	This option changes the host id of the training device. The default is 33.
EntityId	This option changes the entity id of the training device. The default is

	27.
TimeOut	This option will turn on the timing out mechanisms for objects that are orphaned (i.e. ghost entities) in the HLA network. The default is 0.
TimeOutInterval	This option takes as an integer the time in seconds that should pass before timing out an orphan object on the HLA network. The default is 30 seconds.

6.3.3.2.2 Model Mapping Files

The HLA-IU reads two database files that are used to map network model data to host model data.

6.3.3.2.2.1 HLASOSVISMAP.TXT

The file *hlasosvismap.txt* maps the DIS enumeration of a network entity to a model number and emitter model number for the host.

6.3.3.2.2.2 HLAFILTER.TXT

The file *hlafilter.txt* is available to the user to insert DIS enumerations for host filtering. Under configuration management, the file is empty.

6.4 STRICOM Gateway

The STRICOM Gateway version 3.2+ was utilized to allow ModSAF and other DIS native devices to interoperate with the HLA AC-130U. While the gateway proved to be robust in its design, many lessons were learned throughout integration and some issues/anomalies were discovered and resolved.

6.4.1 Gateway Setup

The STRICOM Gateway is ideally installed on a workstation with two ethernet cards, with one card designated for DIS traffic and the other designated as the HLA network card.

Some configuration issues must also be set before the gateway can be used. These options are set in the Sim.cfg file, as explained in the Gateway User Guide. These options must be set to specify site, host, DIS exercise ID, and UDP ports. The DIS IP address is set here, and the network interface card used for the DIS network is also specified here.

Aside from the standard HLA network configuration, i.e., setup of host tables and RTI .rid file, it is important to note the requirements of the Gateway regarding the .fed file. While the RTI uses the .fed file, it is also utilized by the Gateway for initialization functions. Because of this, all classes defined within the provided Gateway .fed file (Giant.fed which reflects the RPR-FOM) must be included within the federation .fed file, regardless of whether or not these classes are part of the FOM. Thus, the federation .fed file was created by utilizing the Gateway .fed file as a baseline and adding additional classes required by VR-Link.

6.4.2 Implementation Issues / Lessons Learned

Please reference Volume I of this report for prior (phase I) implementation issues and lessons learned. All phase I issues were resolved with the delivery and installation of version 3.2 of the Gateway. Additional Gateway related lessons learned are found in section 6.9, Emissions in HLA.

6.5 MetaVR Stealth

The MetaVR system purchased is a PC based single channel view image generator system which supports both DIS and HLA. By specifying the SOF .fed file at load time, the stealth viewer is able to join the SOF federation. Due to a bug in RTI1.3v6, the stealth currently utilizes a model subscription list file to specify the full leaf node specification of entities to which it subscribes.

6.6 MäK Technologies Data Logger

The MäK Technologies Data Logger allows for the recording and playback of DIS and HLA data. This COTS product does provide any analysis or display capabilities pertaining to the data being recorded or played. By specifying the SOF .fed file at load time, the data logger is able to join the SOF federation. The data logger subscribes to all object attributes and interaction parameters specified in the .fed file being used.

6.7 ASTI Radio HLA Compliance

The ASTi Digital Audio Communication System (DACS) of the AC-130U resides on the DIS network and communicates voice information through radio PDUs. As part of the requirements defined within the SOW, this native DIS device was to become HLA compliant as part of this effort. However, as the DACS system is a COTS product and a native HLA solution was not available, the gateway approach to HLA compliance was used. The STRICOM Gateway was used to convert DIS transmitter and signal PDUs to HLA transmitter objects and signal interactions. The radios remained DIS compliant, and the functionality and operation of the radios remain the same.

To verify that radio communications are possible within HLA and to investigate the limitations of using HLA and gateways for radio communications, a test configuration was developed as shown below in Figure 3. The configuration utilizes two ASTi radios operating on isolated DIS networks, and two dual ported STRICOM gateways. DIS radio traffic from one radio is converted into HLA objects and interactions by the first gateway. These are then converted back to DIS by the second gateway.

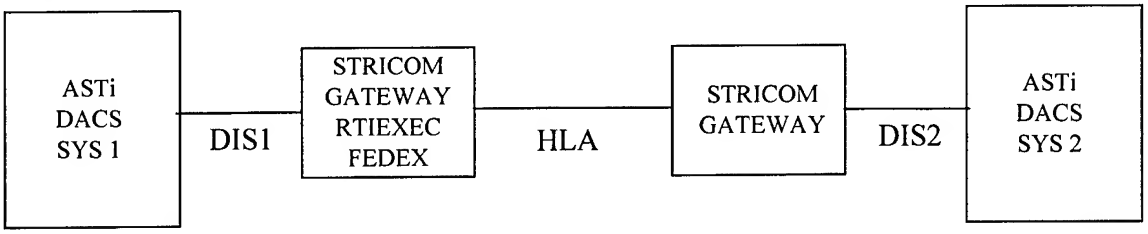

Figure 5 HLA Radio Test Configuration

Several experiments were conducted to determine the performance of the DACS/gateway combination. A simple radio model was created on each DACS, and radio communication was established between the two radios using the above configuration. The number of DIS radios was then increased within each model, and latency measurements were recorded. The experiment continued until an upper limit on the number of transmitting radios was reached.

The results verified that radio communications within HLA are possible with the above gateway approach, but limitations exist. The maximum number of transmitting radios was found to be approximately 12, with additional radios incurring significant break-up in reception. It was not certain as to whether the RTI or the gateway (or a combination of the limitations of both) caused the break-up for additional radios. Latency measurements revealed that an additional 5-10ms was added to the traditional DIS latency between radios, for a total delay of approximately 85-90ms. As more than 12 radios were transmitting during this experiment, the latency began to increase and packet loss became apparent. Note, the experiment was conducted using both best effort and reliable communications as the data transport mechanism, with nearly identical results obtained with each.

Some additional limitations/issues were uncovered as a result of the HLA radio communications study using the STRICOM Gateway. It was found that the gateway did not properly handle PDU concatenation, requiring the ASTi radio to be set to Single PDU mode. In addition, the gateway persists radio objects instead of timing them out. Thus, if a radio was removed from the simulation and another was added with the same frequency as the removed radio, the ASTi radios would not permit radio communications on that frequency. While these limitations are worth noting, they could be resolved through design changes to the STRICOM Gateway.

While the above approach verified radio communications within HLA, the actual AC-130U radio was not officially certified by DMSO as being HLA compliant. Since the radio utilizes the STRICOM Gateway as its interface to the RTI, it is essentially a separate federate from the AC-130U, and hence was not included as part of the AC-130U compliance test.

Until native HLA radio solutions are available, the same configuration used in the experiment above could be used to support long-haul radio communications with a limited number of radios. While it is conceivable that the same gateway that is used for ModSAF (and other DIS tools) could be used to support radio communications, it may be desirable to obtain a

dedicated gateway for each DACS system. From our experiments, it is evident that radio traffic poses increased bandwidth and processing issues, and is best isolated if possible.

6.8 Radios in HLA

Several options exist for incorporating radio voice communications into HLA compliant simulator systems. These options will vary depending on availability of commercial, native HLA radio solutions and results of case studies evaluating voice communication performance within HLA.

For legacy DIS radio systems, an option exists to leave all radio communications as DIS. These radios could be operated on the same network as HLA traffic or be potentially isolated on a dedicated "voice" network. There are definite drawbacks to this approach, however. With either case, if the radio is designed to attach to remote entity objects (utilizing DIS Entity State PDUs), this will no longer be possible since Entity State information is now captured as HLA attributes. Mixing DIS traffic with HLA poses no foreseeable problems, but can congest the network and make trouble shooting difficult. Operating the DIS radios on a separate "voice" network incurs additional cost and logistics, but allows for similar DIS performance. In addition, leaving the radios as pure DIS will not allow the simulation developer to take advantage of HLA features such as Data Distribution Management (DDM).

Several options exist for converting radio communications into HLA. These include the native HLA approach and the gateway approach. Using the native HLA approach, the RTI ambassador is hosted on the radio system itself or as part of the simulator host computer. If the RTI is hosted on the radio, the radio will essentially be considered a separate federate (from the simulator) within the federation. If the RTI is hosted within the host computer, such that radio communications are also handled through the host, then the radio may be treated as an attached system to the simulator.

Another option is to convert DIS radio communications to HLA through the use of a gateway, as described in the previous section. This approach has obvious performance issues including latency and a radio limit, but otherwise is a cost effective alternative. Note, if it is deemed through further experimentation that the performance problems are relating solely to gateway processing, this situation can be resolved by using a dedicated gateway for each radio federate within the federation. Using this approach it is also possible to add frequency filtering to each gateway to receive and filter only those frequencies the radio is interested in.

The HLA compliant radios may also be operated on a dedicated "voice" network, as discussed above. However, if entity state attributes are separated from the voice network, this design will preclude the ability for radios to attach to remote entities. In addition, while radio performance is expected to improve with a dedicated network, unless the simulator radios operate using a completely different federation, performance issues relating to the RTI may still exist.

Whether a native HLA radio or gateway solution is employed, the new services added by the RTI add potentially significant radio capabilities. Declaration Management services may permit interest management filtering for different types of radio communication, thereby

reducing unnecessary radio traffic and host processing. The biggest advantage to using HLA for radio communications relies on the use Data Distribution Management capabilities, available for use with RTI version 1.3 and later. Data Distribution Management may allow filtering based on routing space conditions such as location and frequency.

6.9 Emissions in HLA

Converting Electromagnetic Emission Protocol Data Units (Emission PDU's) from DIS to HLA proved to be a difficult task for our effort. In DIS, a single platform that contains multiple emitter systems and beams sends out all the information in a single Emission PDU. In contrast, HLA separates the different systems and beams that belong to a single platform and creates multiple objects to represent the emission data.

The STRICOM Gateway separates each Emission PDU into multiple objects, and maintains a connection among them to perform the reverse.

Earlier (phase I) Gateway problems associated with emissions were corrected with the implementation of version 3.2 of the Gateway. A few critical items pertaining to Gateway version 3.2 emissions should be noted however. It has been found that the Gateway will not produce emission PDU data unless either the HLA beam parameter index or ERP attribute is non zero. If both are zero then the Gateway interprets this to mean that the beam is in active and does not produce an emission PDU. The AC-130U federate publishes multiple emission systems. The Gateway will construct a complex emission PDU containing all emitter systems and emitter beams for the AC-130U federate. It is critical that the emitter number attribute be a sequential value starting from one for each emitter beam published by the AC-130U federate. The Gateway relies on these values in order to "stack" the emitter systems properly within the complex emission PDU. If, for example, the emitter number attribute was 1 for each system, then the HLA data would all be written into the same first "slot" and only one system would be found in the emission PDU generated.

6.10 HLA Compliance Testing

The HLA compliance testing process has been established as the means to insure DoD simulations are, in fact, HLA-compliant in accordance with DoD policy. The HLA certification process is facilitated through an on-line web-based interface and includes four steps, culminating with the receipt of the HLA compliance certificate. These four steps include the following: 1) Submit a Test Application, 2) Provide a Conformance Notebook, 3) Provide Test Environment Information, 4) Perform Interface Test and Certification Summary Report.

Step 1 includes the submittal of a test application, which consists of the federate developer and sponsor point of contact data, federate name, RTI version, federate description, and expected interface test date. Upon submission of this data, the AC-130U simulator was assigned the test request number 1998-0030.

Step 2 of the certification process includes submitting contents of the Conformance Notebook, including the SOM, Conformance Statement, Scenario data, and FEPW. The AC-130U SOM was originally generated using the OMT.

The Conformance Statement is a list of all RTI services supported by the federate under test. The AC-130U conformance statement was generated by investigating the services utilized by VR-Link, and also the additional federation management services used for simulation pause/resume. A copy of the AC-130U conformance statement is included in the conformance notebook of Appendix G. The conformance statement and SOM are used by the certification agent to perform the conformance cross-check, which verifies that there exist ample objects, attributes, and interactions to support the services identified within the conformance statement.

The Federation Execution Planner's Workbook (FEPW) is utilized by the certification agent to obtain environment and network information. For this effort, only the Summary, Hosts, LANs, and RTI Services tables were necessary and were submitted as part of Step 2.

Step 3 involves the submittal of Test Environment Information to the certification agent, which is used to set up the test logging system. This information includes Host and RTI machine information, the .fed file used, and the RTI .rid file. The RTI .rid file was modified as required for certification test logging. This included adding the variable SERVICE_LOGGING (and setting this to ON) in the .rid file. Upon turning service logging on, the MaxHandleValuePairs value specified within the .rid file needed to be increased to 20,000 (from its default value of 1,000). Otherwise, the AC-130U application would core dump for MaxHandleValuePairs less than 20,000.

Step 4 of the certification process involves executing the Interface Test. The test sequence generated after Step 2 was analyzed, and a procedure was developed to conduct the test. The procedure consisted of a step by step process used to invoke the required RTI services and parameters. The required RTI services included two varieties – calls made by the AC-130U application to the RTI, and RTI callbacks made to the AC-130U. The procedure utilized the STRICOM Gateway (in conjunction with ModSAF) to stimulate the appropriate test conditions that invoked the required service calls.

The interface test was conducted by AB Technologies, a DMSO contractor beginning on November 3, 1998 at the BMC, Hurlburt Field. The STRICOM Gateway was used as the Test Federate and the AC-130U HLA-IU as the Federate Under Test (FUT), as illustrated in the certification test configuration of Figure 4. The DMSO testing tool was used by the certification agent to log all interactions between the RTI and the AC-130U HLA-IU. A log file was generated that is used by a post-processor tool to determine whether the required service calls were successfully invoked and received.

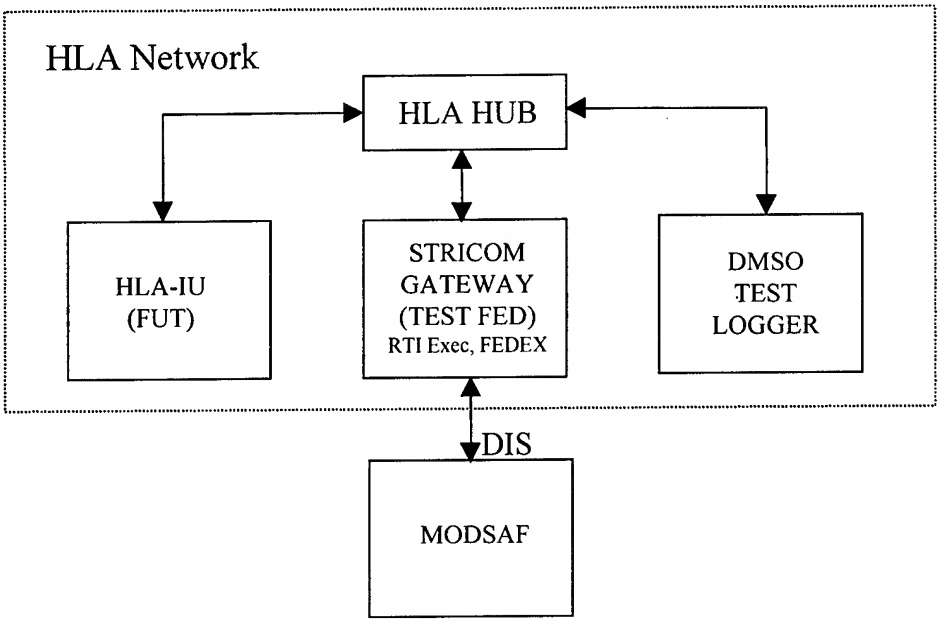

Figure 6 HLA Certification Test Configuration

All required service calls identified within the test sequence were processed properly, and the AC-130U federate passed the HLA certification test on November 4, 1998. Following this step, the certification summary report was applied for, and an after action review was held to discuss feedback on the testing process. A time-line of the entire HLA conformance testing process is illustrated in Table 3.

These steps were essentially repeated for HLA specification 1.3 compliance testing which took place on 22 June, 1999.

Table 2 HLA Conformance Testing Milestones

Compliance Certification Milestone	**Step Completed**
July 2, 1998	Submitted Test Application
July 6, 1998	Application Approved Test ID Assigned User Account Assigned
October 2, 1998	SOM Submitted Scenario RepSOM Submitted Conformance Statement Submitted

	Passed Conformance Cross-check Received Test Sequence File
October 12, 1998	Conformance Statement Resubmitted Fed File Submitted RID File Submitted
October 13, 1998	SOM Resubmitted FEPW Submitted RepSOM Resubmitted New Test Sequence File Received
November 4, 1998	Passed Interface Test
November 11, 1998	Certification Summary Report Requested
November 24, 1998	After Action Review
January 12, 1999	Certification Summary Report Received
June 22, 1999	Passed HLA Specification 1.3 Compliance Test

6.11 On-site Integration

Please reference volume I for phase I implementation notes.

Phase II integration consisted primarily of the following tasks:

- Integration of MäK Technologies Data Logger
- Integration of MetaVR Stealth Viewer
- Continued refinement and enhancement of the HLA-IU
- Understanding of continued networking issues, 7.2.1 C Compiler issues and SCRAMNet memory initialization issues
- Investigation and implementation of modifications required to allow the HLA-IU and DIU to be run from the same disk pack. This was a new site requirement to facilitate to continued use of all previous DIS capabilities should they be required.
- Efforts to understand and comply with developing site configuration management processes and documentation requirements.

The following is a list of lessons learned during the phase II system integration period:

Network: The network setup on site proved to be restrictive and unforgiving. The lack of an expert on the configuration of the CISCO switch/router left us to perform many trials in order to accomplish our goals.

Every host on the network used a 100-MB Phobos Ethernet Card to connect through the CISCO switch/router, which was partitioned as two separate hubs. One partition is for DIS systems, and the other is for non-DIS systems. When we introduced the 10MB Ethernet Card connection from the dual ported HLA gateway into the CISCO switch/router, we discovered that the operating system was hard coding the type of network connection. As a result, we had to find an obscure command called "/usr/bin/FEControl" to auto-detect the type of Ethernet Card being used in the port to get the network connection to work.

Although we succeeded with auto-detecting the type of network card connection, the success was short lived. The CISCO switch/router began to fail, gaining and losing connectivity at indeterminate times. We concluded that the CISCO switch, which was labeled as a "smart-hub", was not smart enough to handle 100-MB and 10-MB Ethernet connections on the same hub. Our solution was to swap our HLA gateway connections. That is, we removed our 10-MB Ethernet connection from the CISCO router, placed it on the mini-hub, and inserted the 100-MB Ethernet connection into the CISCO router.

The constant insertion and removal of the ethernet cables caused one of our connections to become loose. We concluded that a more permanent network setup is warranted in the future. For now our solution was to replace the cable.

Compiler: Compiler issues were solved only after consulting with more than eight Silicon Graphics Inc. technical support calls. Solutions from each call introduced other problems.

Upgrading compilers while retaining an older operating system introduces the risk of having old drivers that won't work with the new compiler. As a lesson, it was important to inform SGI technical support of all the details about the system, so that by chance they may remember some obscure, needed patch.

SGI distributes all compilers as front end compilers, and packages all common files among all types of compilers (FORTRAN, C, C++) as back end compilers. When we upgraded the C++ compiler, we received the front-end compiler and a new back-end compiler for version 7.2.1. This installation made the back-end version 6.2 obsolete. As a result, we were no longer able to compile C source code unless we also upgraded the C compiler to version 7.2.1.

While efforts were made to confirm that the DIU could be recompiled using the 7.2.1 C++ compiler, the SCRAMNet kernal was overlooked. The SCRAMNet kernal is a library which is linked into the DIU and HLA-IU and is made from C source code. Attempts to rebuild the SCRAMNet kernal using the 7.2.1 C compiler indicate that a problem exists with the driver makefile. Communication with SYSTRAN has not yet yielded a solution and direct SGI technical assistance is complicated by the fact that site no longer maintains an SGI technical assistance contract. Suitable workarounds are available in the event that the SCRAMNet

kernal for some reason should require modifications and rebuild. The SCRAMNet kernal can be built from the SOS or MXCI SGIs and then transported to the DIU disk for linkage into the DIU and HLA-IU applications.

SCRAMNet: We uncovered serious dependencies on the DIU that are being overlooked in DIS.

When the ScramNet data structure is changed in the host file *diuds.h*, the source file *hla_scramnet.h* in the /usr/people/sofhla/hlaiu directory also needs to be changed. This is a significant dependency that may be overlooked when the changes are made to the DIU baseline without making them to the HLA-IU baseline.

We discovered another dependency on the ScramNet data structure. The file *hla_init_mem.cc* also needs to be updated when new data structures are added to *diuds.h*. This is because the source file zeroes out each data structure by making a direct reference to each data structure. We eliminated this dependency by taking a pointer from the beginning of the ScramNet memory map and zeroing out each byte through to the end of the ScramNet memory map.

It was subsequently discovered through analysis of inconsistent emission data from the host that initializing the entire SCRAMNet data structure on HLA-IU startup was causing problems. For emission data at least, the Host at initialization writes static data into the emission fields of the SCRAMNet data structure. When the HLA-IU was then started and all SCRAMNet fields initialized to zero, all emission data from the host was lost. DIU SCRAMNet initialization process was then re-examined and duplicated in the HLA-IU.

Operating System: A keyboard must be attached to a host computer, or the boot script will halt without notice.

7. Conclusion

While the Special Operations Forces have achieved a significant milestone in the HLA migration of this training device, the benefit of the new technology will become more evident as other SOF devices are migrated to HLA. The HLA will pave the way for the Special Operations Forces to fully realize the potential of distributed simulation technology, allowing them to cost effectively develop complex simulation applications.

8. Points of Contact

ADST II SOFHLA Team

Gene Lowe	Project Director	407-306-4612
John Bray	Project Engineer	407-306-3509
Gil Gonzalez	Project Engineer	407-306-4604
Bill Garbacz	Systems Engineer	407-306-2310

Ivan Carbia	Software Engineer	407-306-4047

STRICOM

Tom Smith	SOF Project Director	407-384-3666
Robert Miller	Project Director	407-384-3685

NAWCTSD

Víctor Colón	Project Engineer	407-306-4023

Customer Points of Contact

Maj. Mike Vaughn	19th SOS, Hurlburt Field	850-881-2286

Hurlburt Field Site Contacts

Battle Management Center (BMC)		850-581-5339
Scott Kemp	S/W CM	850-581-0040
Dave Sawyer	Tech Support Mgr.	850-881-2368
Sherri Chambers	Eng. Support	850-581-4453
Neil Brennan	Hurlburt Security	850-581-4526

TGDS

Mahesh Patel	Software Engineer	407-306-4007

MäK Technologies Contacts

Len Granowetter	VR-Link S/W lead	617-876-8085 ext. 121
Vrlink-tech-spt@mak.com		
Marc Schlackman	VR-Link Sales	617-876-8085 ext. 131

STRICOM Gateway

Doug Wood	S/W Lead	407-658-5066

Silicon Graphics

Technical Assistance Center		1-800-800-4SGI
Jim Smith	Ft. Walton Rep.	850-651-9760
Al Davis	Tallahassee Rep.	850-580-1404

DMSO Support

Miguel Salazar	Compliance Testing	703-998-1619

rti_support@msis.dmso.mil

9. Acronym List

ADST II	Advanced Distributed Simulation Technology II
BMC	Battle Management Center
CDF	Core DIS Facility
CGF	Computer Generated Forces
CM	Configuration Management
COTS	Commercial Off-the-Shelf
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling & Simulation Office
DO	Delivery Order
DoD	Department of Defense
FEPW	Federation Execution Planner's Workbook
FOM	Federation Object Model
GUI	Graphical User Interface
HIDE	HLA Integrated Development Environment
HLA	High Level Architecture
HLA-IU	HLA Interface Unit

IPT	Integrated Product Team
LAN	Local Area Network
ModSAF	Modular Semi-Automated Forces
OMDT	Object Model Development Tool
OMT	Object Model Template
OSF	Operational Support Facility
PDU	Protocol Data Unit
PVD	Plan View Display
RPR-FOM	Real-Time Platform Reference FOM
RTI	Runtime Infrastructure
SGI	Silicon Graphics, Inc.
SOF	Special Operations Forces
SME	Subject Matter Expert
SMOC	Simulation Middleware Object Classes
SOM	Simulation Object Model
SOW	Statement of Work
STRICOM	Simulation, Training, and Instrumentation Command
WMD	Weapons of Mass Destruction

APPENDIX A

Related Documents in Configuration Management

The following items have been placed in ADST II Configuration Control

DO 81 CDRL

CDRL AB02 – HLA for the AC-130U DO-81,
Final Report and Implementation Results, Volume I
ADST-II-CDRL-HLAAPPS-9800334

CDRL AB02 – HLA for the AC-130U DO-81,
Final Report and Implementation Results, Volume II
ADST-II-CDRL-HLAAPPS-2000085

CDRL A00C – HLA for the AC-130U DO-81,
Simulator Emulator Version Description Document (VDD)
ADST-II-CDRL-HLAAPPS-2000063

CDRL A00C - AC-130U BMC Testbed DO-002 CDRL AB12,
Version Description Document (VDD)
ADST-II-CDRL-AC130U-9700018-D

CDRL A00A – AC-130U BMC Testbed DO-002 CDRL AB14,
Cold Start Procedures Manual (CSPM)
ADST-II-CDRL-AC130U-9700031-E

DO 81 non CDRL

AC-130U BMC Testbed DO-002 CDRL AB10, Interface Design Description (IDD)
ADST-II-CDRL-AC130U-9600136-E

AC-130U NAV/FCO Testbed DO-002 CDRL AB21, Recommended Spare Parts List
ADST-II-CDRL-AC130U-9700025-C

AC-130U BMC Testbed DO-002 CDRL AB02, System Performance Specification (SPS)
ADST-II-MISC-009R-9600183-G

AC-130U BMC Testbed DO-002 CDRL AB11, Software User Manual (SUM)
ADST-II-CDRL-AC130U-9700030-E

AC-130U BMC Testbed DO-002 CDRL AB06, System Segment Design Description
ADST-II-CDRL-AC130U-9700015-F

AC-130U BMC Testbed DO-002 CDRL AB08, Software Requirements Specification
ADST-II-CDRL-AC130U-9700017-D

AC-130U BMC Testbed DO-002 CDRL AB02, Network Implementation Plan
ADST-II-CDRL-AC130U-9600152-B

HLA for the AC-130U DO-81, AC-130U Simulation Object Model (SOM)
ADST-II-MISC-HLAAPPS-2000065

HLA for the AC-130U DO-81, Network User's Manual (NUM)
ADST-II-MISC-HLAAPPS-2000064

HLA for the AC-130U DO-81, HLA-IU Engineering Unit Test Procedures
ADST-II-MISC-HLAAPPS-2000068

HLA for the AC-130U DO-81, High Level Architecture Systems Training (presentation)
ADST-II-MISC-HLAAPPS-2000067

HLA for the AC-130U DO-81, HLA Interface Unit Acceptance Test Procedure (ATP)
ADST-II-MISC-HLAAPPS-2000084

HLA for the MC130E/MC130H Combat Talon DO-105,
Special Operations Forces (SOF) Federation Object Model (FOM)
ADST-II-MISC-MC130-2000025

APPENDIX B

FOM Variances between the SOF FOM 0.5 and RPR-FOM 0.5

The SOF FOM 0.5 utilized the RPR-FOM 0.5 as a baseline, and extensions were added to support key functionality required by the SOF simulator devices. At the time of this effort, the required functionality did not presently exist within the RPR-FOM. The following defines the differences between the SOF FOM and RPR-FOM 0.5:

- **Addition of IFF** – Required for publication by the MC-130E and MC-130H. IFF is a DIS 1278-1998 capability, which will be incorporated into RPR-FOM 2.0, and was not available in RPR-FOM 0.5. IFF was implemented in the SOF FOM 0.5 as a hierarchy of object classes under the Embedded System class, including an IFF abstract class, IFFInterogator, IFFTransponder, and IFFOther class. Each object class contains all attributes necessary to implement IFF for the combat talons. It is important to note that the STRICOM Gateway interface is designed to interoperate with this implementation of IFF, and will generate the appropriate IFF PDU.
- **Light States** – Required by MH-47E and MH-60K simulators. RPR-FOM 0.5 denoted LightsState as an enumerated attribute under the Physical Entity class, but did not define these enumerations within the enumerated datatype table. Consequently, it was decided to implement all light states as separate attributes in the SOF FOM 0.5, where each attribute defines the binary state of the respective light. These light state attributes were defined within their respective platform classes. For example, BrakeLightsOn is an attribute defined within the MilitaryLandPlatform class, and FormationLightsOn is an attribute defined within the MilitaryAirLandPlatform class. It is important to note that RPR-FOM 1.0 addressed the lack of light state definition, and chose to implement light states as individual attributes under the Platform Class (in favor of leaving the leaf object classes attributeless). Future versions of the SOF FOM should follow suit to the RPR-FOM 1.0 definitions of light states.

APPENDIX C

HLA compliance Testing

C 2. Conformance Statement
C 5. Testable Sequence
C 7. Certification Summary Report

Please note that the following sections Are hardcopy only:
C 7.

APPENDIX C1 - Conformance Statement ADST-II-HLAAPPS-2000085
April 3, 2000

createFederationExecution YES
destroyFederationExecution YES
joinFederationExecution YES
resignFederationExecution YES
registerFederationSynchronizationPoint NO
confirmSynchronizationPointRegistration NO
announceSynchronizationPoint NO
synchronizationPointAchieved NO
federationSynchronized NO
requestFederationSave NO
initiateFederateSave NO
federateSaveBegun NO
federateSaveComplete NO
federationSaved NO
requestFederationRestore NO
confirmFederationRestorationRequest NO
federationRestoreBegun NO
initiateFederateRestore NO
federateRestoreComplete NO
federationRestored NO
publishObjectClass YES
unpublishObjectClass NO
publishInteractionClass YES
unpublishInteractionClass NO
subscribeObjectClassAttributes YES
unsubscribeObjectClass NO
subscribeInteractionClass YES
unsubscribeInteractionClass NO
startRegistrationForObjectClass YES
stopRegistrationForObjectClass YES

turnInteractionsOn YES
turnInteractionsOff YES
registerObjectInstance YES
discoverObjectInstance YES
updateAttributeValues YES
reflectAttributeValues YES
sendInteraction YES
receiveInteraction YES
deleteObjectInstance YES
removeObjectInstance YES
localDeleteObjectInstance NO
changeAttributeTransportationType NO
changeInteractionTransportationType NO
attributesInScope NO
attributesOutOfScope NO
requestAttributeValueUpdate YES
provideAttributeValueUpdate YES
turnUpdatesOnForObjectInstance YES
turnUpdatesOffForObjectInstance YES
unconditionalAttributeOwnershipDivestiture NO
negotiatedAttributeOwnershipDivestiture NO
requestAttributeOwnershipAssumption NO
attributeOwnershipDivestitureNotification NO
attributeOwnershipAcquisitionNotification NO
attributeOwnershipAcquisition NO
attributeOwnershipAcquisitionIfAvailable NO
attributeOwnershipUnavailable NO
requestAttributeOwnershipRelease NO
attributeOwnershipReleaseResponse NO
cancelNegotiatedAttributeOwnershipDivestiture NO
cancelAttributeOwnershipAcquisition NO
confirmAttributeOwnershipAcquisitionCancellation NO
queryAttributeOwnership NO
informAttributeOwnership NO

isAttributeOwnedByFederate NO
enableTimeRegulation NO
timeRegulationEnabled NO
disableTimeRegulation NO
enableTimeConstrained NO
timeConstrainedEnabled NO
disableTimeConstrained NO
timeAdvanceRequest NO
timeAdvanceRequestAvailable NO
nextEventRequest NO
nextEventRequestAvailable NO
flushQueueRequest NO
timeAdvanceGrant NO
enableAsynchronousDelivery NO
disableAsynchronousDelivery NO
queryLBTS NO
queryFederateTime NO
queryMinimumNextEventTime NO
modifyLookahead NO
queryLookahead NO
retract NO
requestRetraction NO
changeAttributeOrderType NO
changeInteractionOrderType NO
createRegion NO
modifyRegion NO
deleteRegion NO
registerObjectInstanceWithRegion NO
associateRegionForUpdates NO
unassociateRegionForUpdates NO
subscribeObjectClassAttributesWithRegion NO
unsubscribeObjectClassWithRegion NO
subscribeInteractionClassWithRegion NO
unsubscribeInteractionClassWithRegion NO

```
sendInteractionWithRegion NO
requestAttributeValueUpdateWithRegion NO
enableClassRelevanceAdvisorySwitch YES
disableClassRelevanceAdvisorySwitch NO
enableAttributeRelevanceAdvisorySwitch YES
disableAttributeRelevanceAdvisorySwitch NO
enableAttributeScopeAdvisorySwitch NO
disableAttributeScopeAdvisorySwitch NO
enableInteractionRelevanceAdvisorySwitch YES
disableInteractionRelevanceAdvisorySwitch NO
```

APPENDIX C2

TESTABLE SEQUENCE FILE

Testable Sequence

Below is a list of all the services that must be used during
IF Testing
Note, the format of each service is as follows:
    ServiceName(Object or Interaction, Attribute or Parameter)

If there are no parameters shown, it is assumed that
1. No parameters necessary
   or
2. Any parameter is valid

deleteObjectInstance
discoverObjectInstance
discoverObjectInstance (MilitaryAirLandPlatform)
enableAttributeRelevanceAdvisorySwitch
enableClassRelevanceAdvisorySwitch
enableInteractionRelevanceAdvisorySwitch
joinFederationExecution
provideAttributeValueUpdate
publishInteractionClass
publishInteractionClass (StartResume)
publishInteractionClass (StopFreeze)
publishObjectClass
publishObjectClass (MilitaryAirLandPlatform, DRAlgorithm)
publishObjectClass (MilitaryAirLandPlatform, EntityID)
publishObjectClass (MilitaryAirLandPlatform, ForceID)
receiveInteraction
receiveInteraction (StartResume)
receiveInteraction (StopFreeze)
reflectAttributeValues
reflectAttributeValues (DRAlgorithm)
reflectAttributeValues (EntityID)
reflectAttributeValues (ForceID)
registerObjectInstance
registerObjectInstance (MilitaryAirLandPlatform)
removeObjectInstance
requestClassAttributeValueUpdate
requestObjectAttributeValueUpdate
resignFederationExecution
sendInteraction
sendInteraction (StartResume)
sendInteraction (StopFreeze)
startRegistrationForObjectClass
stopRegistrationForObjectClass
subscribeInteractionClass
subscribeInteractionClass (StartResume)
subscribeInteractionClass (StopFreeze)

```
subscribeObjectClassAttributes
subscribeObjectClassAttributes (MilitaryAirLandPlatform, DRAlgorithm)
subscribeObjectClassAttributes (MilitaryAirLandPlatform, EntityID)
subscribeObjectClassAttributes (MilitaryAirLandPlatform, ForceID)
turnInteractionsOff
turnInteractionsOn
turnUpdatesOffForObjectInstance
turnUpdatesOnForObjectInstance
updateAttributeValues
updateAttributeValues (DRAlgorithm)
updateAttributeValues (EntityID)
updateAttributeValues (ForceID)
```